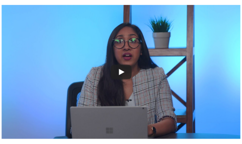
Linear Regression

In our first lab, we're going to use Azure Machine Learning Studio to train a model using one of the fundamental machine learning algorithms: *Linear regression*. Before we dive into the lab, let's review what linear regression is and how it can be used to train a model.

The video below gives a brief review of the main concepts. If you've never seen linear regression before, or need a more thorough review, you can continue on for a detailed explanation below.



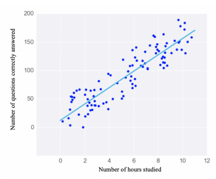
If you feel confident in your understanding of linear regression, feel free to move ahead to the next page and get started on the lab. Otherwise, we'll go over the concepts in further detail below.

Understanding Linear Regression

> *As the term suggests,* ***linear regression*** *is an algorithm that uses a straight line (or plane) to describe relationships between variables.*

Let's consider a very simple example of a linear relationship. Suppose that we want to know if the *number of hours a student spends studying for a test* is related to the *number of questions answered correctly on the test*.

Such a relationship might look something like this:



We can see that there is a clear relationship: Students who spent more time studying also scored higher on the test.

What is more, we can see that the data points cluster around a straight line. Linear regression is all about finding the *line that best fits the data*. And this model (or line) can then be used to make predictions. In this case, if we know the number of hours a student has studied for the test, we can predict how many questions he or she can answer correctly. To make this prediction, we need the equation for the line of best fit. What would that look like?

Simple Linear Regression

You may recall from fundamental algebra that the general equation for a line looks like this:

$y = mx + b$

Where m is called the *slope* of the line, and *b* is the y-intercept. Again, this is the *general* equation. For a specific line, we need to know the values for the slope and y-intercept. For example, the following equations represent three different lines.

$y = 10x + 50$

$y = 2x + 3$

$y = -10x + 40$

Equations like these can be used to make *predictions*. Once we know m and b , we can feed in a value for x and the equation will give us the value of y.

QUESTION 1 OF 2

For the earlier example of students studying for a test, suppose that we find that the line of best fit is:

$y = 15x + 3$

Where x is the number of hours studied and y is the number of questions correctly answered.

If a student studied 10 hours for the test, what is their predicted score?

- 48
- 148
- [✓ 153]
- 183

SUBMIT

QUESTION 2 OF 2

Thinking back to the *models vs. algorithms* distinction, is our equation...

$y = 15x + 3$

...best described as a **model** or an **algorithm**?

- [✓ $y = 15x + 3$ is a model]
- $y = 15x + 3$ is an algorithm

SUBMIT

Linear Regression in Machine Learning

The equation we used above was:

$y = mx + b$

In algebraic terms, we may refer to m as the **coefficient** of x or simply the **slope** of the line, and we may call b the **y-intercept**. In machine learning, you will typically see the y-intercept referred to as the **bias**. In machine learning, you will also often see the equation represented using different variables, as in:

$y = B_0 + B_1 * x$

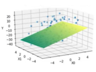
The letters are different and the order has been changed, but it is exactly the same equation. Thus, we can see that what we know from algebra as the basic equation for a line is also, in machine learning, the equation used for **simple linear regression**.

Multiple Linear Regression

In more complex cases where there is *more than one* input variable, we might see something like this:

$y = B_0 + B_1 * x_1 + B_2 * x_2 + B_3 * x_3 ... + B_n * x_n$

In this case, we are using multiple input variables to predict the output. When we have *multiple* input variables like this, we call it **multiple linear regression**. The visualization of multiple linear regression is no longer a simple line, but instead a *plane* in multiple dimensions:



But don't let any of this intimidate you: The core idea is still that we are modeling a relationship (using a line or plane) in order to help us predict the value of some variable that we are interested in.

Training a Linear Regression Model

To "train a linear regression model" simply means to learn the *coefficients* and *bias* that *best fit the data*. This is the purpose of the linear regression algorithm. Here we will give you a high-level introduction so that you understand conceptually how it works, but we will not go into the mathematical details.

The Cost Function

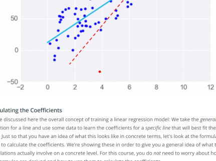
Notice from our example of test scores earlier that the line we came up with did not *perfectly* fit the data. In fact, *most* of the data points were not on the line! When we predict that a student who studies for 10 hours will get a score of 153, we do not expect their score to be exactly 153. Put another way, when we make a prediction using the line, we expect the prediction to have some **error**.

The process of finding the best model is essentially a process of finding the coefficients and bias that *minimize* this error. To calculate this error, we use a **cost function**. There are many cost functions you can choose from to train a model and the resulting error will be different depending one which cost function you choose. The most commonly used cost function for linear regression is the **root mean squared error (RMSE)**

Preparing the Data

There are several **assumptions** or conditions you need to keep in mind when you use the linear regression algorithm. If the raw data does not meet these assumptions, then it needs to be prepared and transformed prior to use.

- **Linear assumption:** As we've said earlier, linear regression describes variables using a *line*. So the relationship between the input variables and the output variable needs to be a linear relationship. If the raw data does not follow a linear relationship, you may be able to transform) your data prior to using it with the linear regression algorithm. For example, if your data has an exponential relationship, you can use *log transformation*.
- **Remove collinearity**: When two variables are **collinear**, this means they can be modeled by the same line or are at least highly *correlated*; in other words, one input variable can be accurately predicted by the other. For example, suppose we want to predict education level using the input variables `number of years studying at school`, `if an individual is male`, and `if an individual is female`. In this case, we will see collinearity—the input variable `if an individual is female` can be perfectly predicted by `if an individual is male`, thus, we can say they are highly correlated. Having highly correlated input variables will make the model less consistent, so it's important to perform a *correlation check* among input variables and remove highly correlated input variables.
- **Gaussian (normal) distribution:** Linear regression assumes that the distance between output variables and real data (called *residual*) is *normally distributed*. If this is not the case in the raw data, you will need to first transform the data so that the residual has a normal distribution.
- **Rescale data:** Linear regression is very sensitive to the distance among data points, so it's always a good idea to *normalize* or *standardize* the data.
- **Remove noise**: Linear regression is very sensitive to noise and *outliers* in the data. Outliers will significantly change the line learned, as shown in the picture below. Thus, cleaning the data is a critical step prior to applying linear regression.



Calculating the Coefficients

We've discussed here the overall concept of training a linear regression model: We take the *general* equation for a line and use some data to learn the coefficients for a *specific line* that will best fit the data. Just so that you have an idea of what this looks like in concrete terms, let's look at the formulas used to calculate the coefficients. We're showing these in order to give you a general idea of what the calculations actually involve on a concrete level. For this course, you do *not* need to worry about how the formulas are derived and how to use them to calculate the coefficients.

The formula for getting the slope of the line looks something like this:

$$B1 = \frac{\sum_{i=1}^{n}(x_i - mean(x)) \times (y_i - mean(y))}{\sum_{i=1}^{n}(x_i - mean(x))^2}$$

To get the intercept, we calculate:

$$B0 = mean(y) - B1 \times mean(x)$$

And to get the *root mean squared error (RMSE)*, we have:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - y_i)^2}{n}}$$

In most machine learning libraries (such as Sklearn or Pythorch) the inner workings of the linear regression algorithm are implemented for you. The error and the best coefficients will be automatically calculated when you input the data. Here, the important thing is to understand what is happening conceptually—namely, that we choose a cost function (like RMSE) to calculate the *error* and then *minimize* that error in order to arrive at a *line of best fit* that models the training data and can be used to make predictions.

Now that we've review the concept, let's get some hands-on practice implementing the linear regression algorithm in Azure Machine Learning Studio!